SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V10I9P107

Volume 10 Issue 9, 63-71, September 2023
© 2023 Seventh Sense Research Group®

Original Article

Analysis of Offensive Data over Multi-Source Social Media Environment Using Modified Random Forest Algorithm

V. Uma Maheswari[1], R. Priya[2]

[1]*Department of Computer Science, School of Computing Sciences,*
Vels Institute of Science, Technology & Advanced Studies (VISTAS), Chennai, India.
[2]*Department of Computer Applications, School of Computing Sciences,*
Vels Institute of Science, Technology & Advanced Studies (VISTAS), Chennai, India.

[1]*Corresponding Author : uma219ravi@gmail.com*

Received: 13 July 2023 Revised: 22 August 2023 Accepted: 11 September 2023 Published: 30 September 2023

Abstract *- The widespread usage of social media platforms has resulted in an increasing volume of offensive content, posing significant challenges to maintaining a safe and respectful online environment. This research presents an analysis of offensive data over the social media environment using a modified Random Forest algorithm. The proposed modification to the traditional Random Forest algorithm incorporates a Weighted class Random Forest (WRF) to enhance model diversity and robustness. An algorithm utilizes weighted classes during training to address the inherent class imbalance in offensive data. By assigning higher weights to offensive content, the model prioritizes accurately identifying offensive posts, comments, and messages. This paper used the Twitter and Reddit dataset of multi-source social media content, labeled for offensive and non-offensive content, to train and validate the modified Random Forest model. Our proposed model is compared with Decision Tree (DT), Extreme-Gradient Boosting (XGBoost), Multi-Layer Perceptron (MLP), K-Nearest Neighbors (KNN), and Traditional Random Forest (RF) algorithms in machine learning. A number of performance metrics are used to assess the model's effectiveness in dealing with offensive data, including accuracy, recall, precision, specificity, and the F1-score. The results demonstrate that the modified Random Forest algorithm outperforms better than other machine learning algorithms. Moreover, the model shows improved resilience to variations in offensive language and context, making it more suitable for real-world applications.*

Keywords *- Social media, Offensive data, Content moderation, Machine learning, Modified Random Forest algorithm, Weighted class Random Forest.*

1. Introduction

As social media platforms have evolved, people interact, share information, and have conversations in a whole new way. Communication in the modern world is dominated by social media, offering a platform for individuals, communities, and organizations to connect, express opinions, and disseminate content on a global scale. However, this unprecedented level of connectivity has also given rise to various challenges, one of the most concerning being the proliferation of offensive content on these platforms [1].

Offensive data over the social media environment encompasses a wide range of harmful and inappropriate content, such as hate speech, cyberbullying, harassment, misinformation, and explicit or violent material [2]. The rapid growth of social media users and the ease of sharing content have made it increasingly difficult for platforms to effectively monitor and moderate offensive data [3]. The consequences of unaddressed offensive content can be far-reaching, affecting individuals emotionally, mentally, and sometimes even physically while also fuelling online toxicity and undermining the sense of online community [4]. The scale and complexity of offensive data on social media demand advanced technological solutions to tackle this issue [5]. Traditional content moderation techniques have proven insufficient, as offensive data can often be subtle, context-dependent, and evolving with language trends. Consequently, researchers and technology companies have been actively exploring the application to identify and mitigate offensive content efficiently and accurately [6].

This research focuses on analyzing offensive data over the social media environment using a modified Random

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

Forest algorithm. The RF algorithm, a powerful ensemble learning technique, has shown promise in various domains for its ability to handle complex data, maintain model diversity, and resist overfitting [7]. RF combines the predictions from multiple decision trees built during the training period to make the final decision. The training is conducted using a bootstrapped sample of the original dataset for each decision tree [8]. A decision tree is further distinguished from traditional decision trees by considering only a random subset of features at each split, which introduces a random element and a sense of diversity [9].

We are developing a model to detect offensive data accurately and efficiently while minimizing false positives and false negatives [10]. By addressing the inherent class imbalance in offensive data through weighted classes during training, the model is designed to prioritize detecting offensive content, allowing social media platforms to take proactive measures to moderate and filter harmful material. In the subsequent sections, we will delve into the details of our modified Random Forest algorithm, the experimental methodology, and the results obtained.

However, several research gaps and challenges still exist in this field. Machine learning models often struggle to understand the context in which offensive language or content is used. Social media content often includes text, images, videos, and audio. Integrating and analyzing multiple data modalities to detect offensive content effectively is a challenging and evolving area of research.

2. Related Works

A substantial amount of the younger generation's time is devoted, both actively and passively, to the use of various forms of social media. The meteoric rise of Open Street Map has ushered in an era marked by an uptick in the frequency of cybercrime [11]. A deep learning system was suggested in this study [12], which would analyze real-time tweets or social media postings and accurately detect any material related to cyberbullying that may be present in such tweets or posts. Recent research has demonstrated that strategies that employ deep neural networks are more successful than traditional methods when it comes to recognizing messages that include instances of cyberbullying. In this technique [13], the likelihood of creating assertive speeches is decreased, and using a few repetitions speeds up the training process while simultaneously reducing the rate at which BLEU values fall. In addition, this method successfully slows down the reduction in BLEU values. The idea of toxicity is defined in this study [14] as a suggested concept based on psychology and social philosophy. Then, presented a strategy that resolves uncertainty across these categories by identifying various dimensions of hazard and incorporating explicit information into the statistical learning algorithm. This approach was developed by us. This research work [15] presents a method for identifying hate speech from social media websites based on a mixture of word-integration techniques.

According to the findings of this research [16], machine learning methods are outperformed by transformer-based models. In addition, adapter-based strategies are superior to adjusted methods with regard to both the amount of time they take and the amount of efficiency they provide when dealing with low-resource languages like Tamil.

Identifying content that is insulting, abusive, or motivated by hatred has become an essential component of online abuse. The manual identification of cyberharassment is laborious, time-consuming, and impossible in environments where data is continually rising. Within the scope of this work [17], we tackled the difficulties associated with the automated identification of abusive tweets written in Arabic.

The present paper [18] outlines some methodological obstacles that must be overcome when developing automated hate crime prevention systems. Our efforts to counteract hostile information on the internet were motivated by the issues we faced in this larger domain. In this study [19], the authors made an effort to illustrate the many different types of abusive behaviour that a person could experience when using the internet and the relevance of being able to recognize them. They did so by dividing these behaviours into four different groups: content-based, sentiments and emotion-based, user or profile-based, and networks or graph-based approaches.

Each iterative process is enhanced with decision trees, the classifier's enactment is monitored, and overfitting is protected through an IRF process [20]. Based on this model, a new decision tree technique is given, in which the level of attribute dependency is employed as a partition measure [21]. Using 14 datasets from UCI's Machine Learning Repository, tested the effectiveness of our approach. This article [22] proposes an enhanced RF for text categorization that integrates bootstrapping and randomized subspace approaches concurrently. The enhanced RF for text classification is dubbed modified for text classification. False detection rates are often high when training intrusion detection models. This is because insufficient training data is available, which is caused by a discrepancy in the data [23].

The RF model showed great experimental results that improved performance, ensured predictive maintenance, and avoided wasting energy with this application [24]. These benefits were proven by the outstanding test results obtained on the random forest model. The Random Forest Classification has been improved, which has led to improved overall performance. The accuracy of the findings acquired from the suggested approach is preferable [25] compared to the precision of the results achieved through the current methods.

Many studies use Natural Language Processing (NLP) techniques, such as sentiment analysis, text classification, and topic modeling, to detect offensive or abusive language in social media posts. Offensive content is not limited to text; it can also include images, videos, and audio. Researchers have developed models to capture context and differentiate between offensive and non-offensive language based on the conversation's context. Analyzing cross-platform trends can help identify offensive content patterns. Addressing bias in offensive content detection models and ensuring fairness in the moderation process is an ongoing challenge.

3. Proposed Model

A proposed methodology on multi-source social media platforms for identifying offensive posts of Modified Random Forest to classify whether the texts or comments posted on social media platforms contain offensive content or not. The data were collected from two different social media platforms, such as Reddit and Twitter. The collected data were pre-processed and deployed into the MRF algorithm and compared DT, MLP, XGBOOST, KNN and RF algorithms in machine learning. An overall proposed framework model is shown in Figure 1.

3.1. Dataset Description

The data were collected from "https://www.kaggle.com/datasets/cosmos98/twitter-and-reddit-sentimental-analysis-dataset". The Twitter and Reddit comments are cleaned using Pythons and NLP, and each is assigned a sentiment label between -1 and 1. A 0 indicates neutrality, a 1 indicates positivity, and a -1 indicates negativity. Sentiment labels have been assigned to around 163K Tweets in a dataset. About 37K comments are included in the Reddit.csv dataset, along with their sentimental labels.

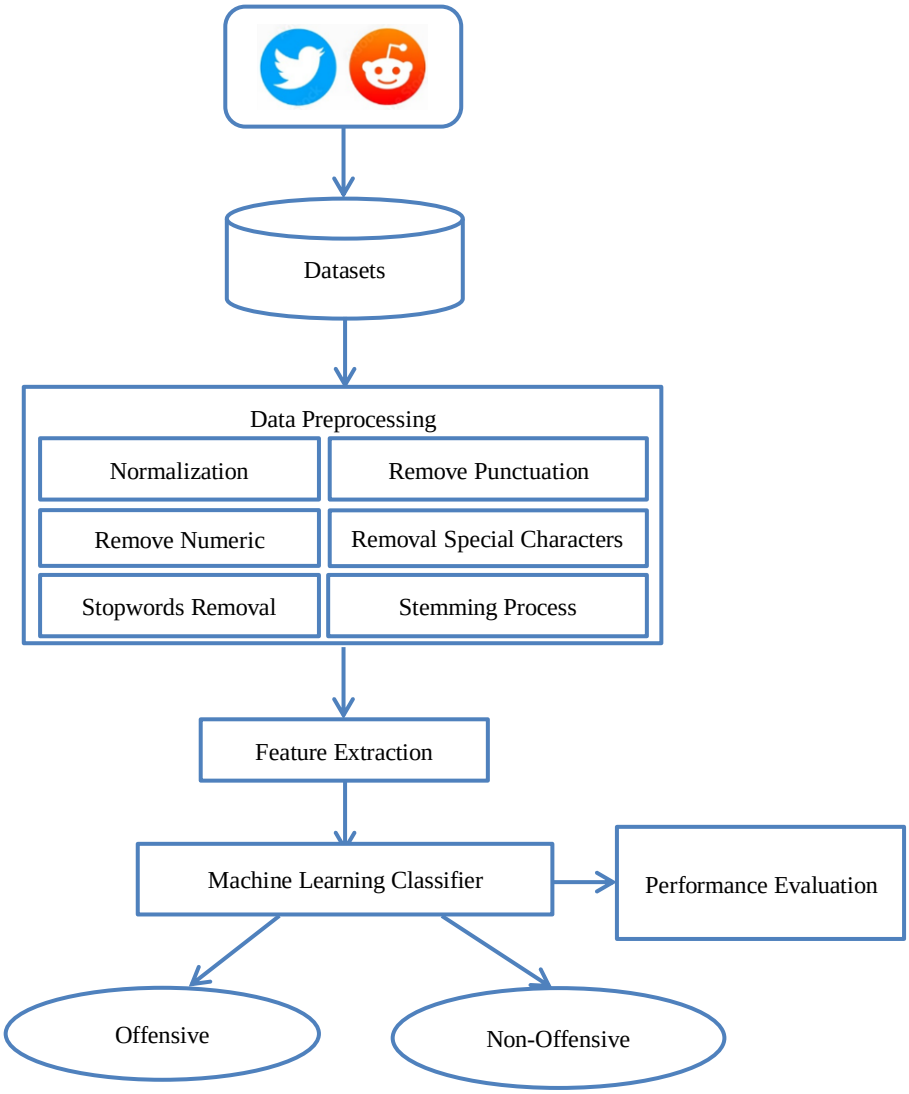

Fig. 1 Proposed framework

3.2. Pre-Processing

Online social media content requires pre-processing prior to being analyzed meaningfully. In this method, raw data is cleaned, transformed, and enhanced so that it can be modeled and analyzed later on. The objective of data pre-processing is to improve the data quality, reduce noise, handle inconsistencies, and create structured representations that facilitate effective data analysis and interpretation.

Text cleaning is a fundamental step in data pre-processing, particularly when dealing with textual data from sources like social media posts, comments, reviews, or articles. Cleansing text removes irrelevant information, noise, and inconsistencies, making it more useful for text analysis and natural language processing. Here are the key text-cleaning techniques commonly used in data pre-processing:

3.2.1. Normalization

Normalization involves converting words to their standard form to reduce the number of variations of the same word. It can include converting text to lowercase, removing diacritics, and handling accent marks.

3.2.2. Lemmatization

By lemmatizing words, we further reduce the dimensionality of text data by reducing them to their root form (lemmas).

3.2.3. Stop-Word Removal

Stop words are common words like "the," "is," "in," "and," etc., that occur frequently in a language but often do not contribute much to the meaning of the text. Removing stop words helps reduce noise and decreases the computational burden during analysis.

3.2.4. Stemming

In stemming, suffixes are removed from a word in order to form the word. Even if the resulting stem is not a valid word, the goal is still to reduce the word to its base form.

3.2.5. Removing Special Characters

Special characters, such as punctuation marks, symbols, and emojis, are typically not meaningful in text analysis. Removing them helps in focusing on the essential words and content.

3.2.6. Removing Numeric

Removing numeric digits from the text can be beneficial, especially when numbers are irrelevant to the analysis, such as sentiment analysis or topic modeling.

3.2.7. Removing Punctuations

Punctuation marks like commas, periods, and question marks are essential for language, but they are often removed during text pre-processing to simplify the analysis and avoid unnecessary distinctions between words.

Pseudocode for Data Pre-processing

```
def preprocess_data(data):
    for column in data.columns:
        if data[column].isnull().any():
            data[column].fillna(value='', inplace=True)
    data = data.drop_duplicates()
    "data['text'] = data['text'].apply(lambda text: text.lower())
    data['text'] = data['text'].apply(lambda text: text.replace('.', ' '))"
    stop_words = set(words)
    return data
```

This pseudocode first checks for missing data in the data set. If any missing data is found, it is filled in with empty strings. Next, the duplicate data is removed from the data set. In the next step, all text data is converted to lowercase, punctuation is removed, and then the data is cleaned, removing stop words. Finally, the pre-processed data is returned.

3.3. Feature Extraction

Feature extraction using Term Frequency-Inverse Document Frequency (TF-IDF) measures the importance of each word in each document based on its position in the corpus. TF-IDF analyzes the importance of each word relative to all the others in the corpus. Documents are represented as vectors, with each dimension corresponding to a unique word in the corpus, and the values of the dimensions are the TF-IDF scores of those words. Here is a step-by-step process for feature extraction using TF-IDF:

3.3.1. Tokenization

Using tokenization, extract individual words or tokens from each document. In this step, the text is broken up into words and n-grams.

3.3.2. Term Frequency (TF) Calculation

The term frequency in a document measures how often words (terms) appear. A word's frequency of appearance in a document is the ratio between the number of words in that document and its total number. In a document, occurrences of words are given higher weight since they are likely to have a higher relevance to the content. Identify each word in each document by calculating its Frequency (TF). An individual word's frequency in a document is determined by dividing it by its total number of words.

$$TF\ (word, document) = (Number\ of\ occurrences\ of\ the\ word\ in\ the\ document) / (Total\ number\ of\ words\ in\ the\ document)$$

3.3.3. Inverse Document Frequency (IDF) Calculation

By measuring the rarity of a word across the entire collection, IDF measures its rarity. This ratio is obtained by

taking the logarithm of total documents to documents containing a particular word. Rather than giving greater weight to common words, we would rather give weight to rare words throughout the collection because they provide better discrimination and differentiation.

All words in the corpus should be calculated for their IDF. By increasing the logarithm of the IDF, the number of documents containing the word increases.

$$IDF\ (word) = log\_e\ (Total\ number\ of\ documents\ /\ Number\ of\ documents\ containing\ the\ word)$$

3.3.4. TF-IDF Calculation

Divide the TF and IDF values by the number of words in each document and calculate the TF-IDF score.

$$TF - IDF\ (word, document) = TF\ (word, document) * IDF\ (word)$$

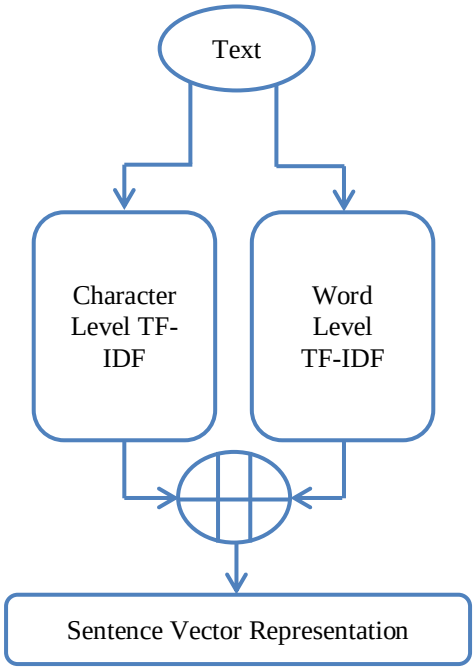

Fig. 2 TF-IDF vectorization process

3.3.5. Vectorization

Represent each document as a vector of TF-IDF scores. As shown in Figure 2, the vectorization process takes place. Dimensions in this vector are based on the TF-IDF scores of each unique word across the corpus. Through the TF-IDF technique, individual words are considered in terms of their rarity within the entire corpus and their importance.

Pseudocode for TF-IDF

```
def tf_idf(documents):
    term_frequencies = {}
  for document in documents:
    for word in document:
      if word not in term_frequencies:
        term_frequencies[word] = 0
      term_frequencies[word] += 1
  idf_scores = {}
  for word in term_frequencies:
    idf_scores[word] = math.log(len(documents) / (1 + term_frequencies[word]))
  tf_idf_scores = {}
  for word in term_frequencies:
    tf_idf_scores[word] = term_frequencies[word] * idf_scores[word]
  return tf_idf_scores
def main():
  documents = [""]
  tf_idf_scores = tf_idf(documents)
```

This pseudocode first calculates the term frequencies for all words in the documents. Then, it calculates the inverse document frequency for each word. Finally, it calculates the TF-IDF scores for all words. The main function takes a list of documents as input and returns a dictionary of TF-IDF scores. The dictionary maps each word to its TF-IDF score.

3.4. MRF Classifier

Weighted class Random Forest is a modification of the traditional Random Forest algorithm that addresses the issue of class imbalance in classification tasks. The classifier can be biased, such that it performs poorly on the minority class while performing well on the majority class when one class has significantly more instances than the others. When classes are weighted during training, the algorithm becomes more sensitive to the minority class and performs better on imbalanced datasets because the minority class is assigned more weights. The main steps in the Weighted class Random Forest algorithm are:

3.4.1. Data Bootstrapping and Tree Building

Randomly sample the training data with replacement to create multiple bootstrap samples (subsets) of the data. Unlike the original dataset, each bootstrap sample may contain one or more subsets, some duplicate instances and exclude others. For each bootstrap sample, build a decision tree using a subset of the features (random feature selection) at each split. As a result, the algorithm places a higher weight on instances of minorities than on instances of majorities during the construction of each tree, as shown in Figure 3.

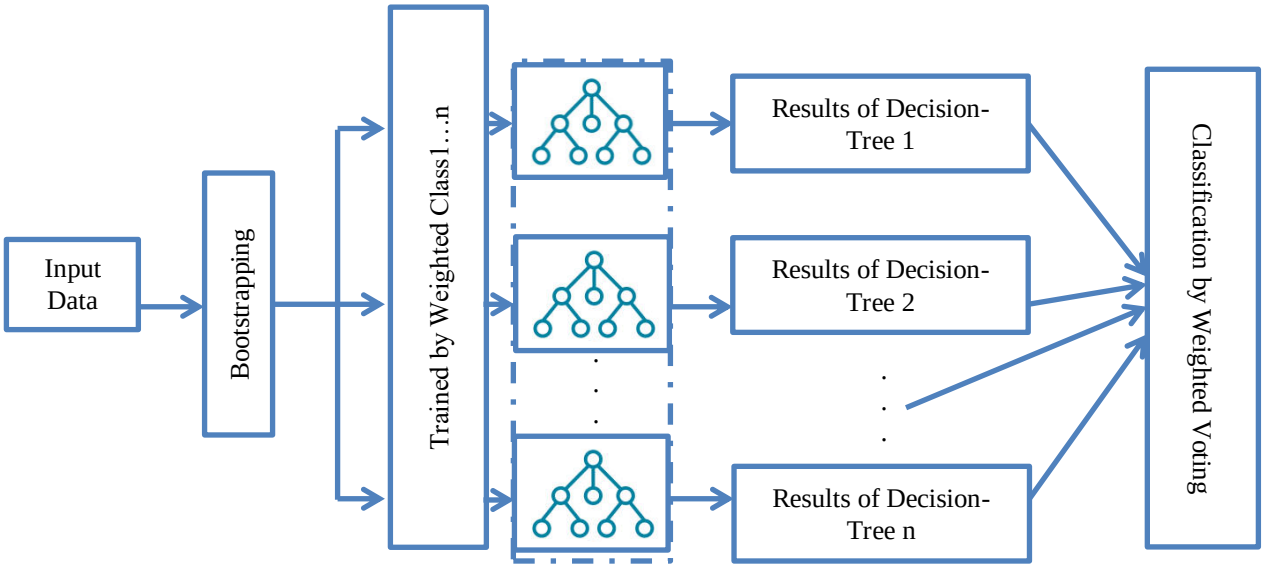

Fig. 3 TF-IDF vectorization process

The steps for data bootstrapping and tree building are the same as in the traditional Random Forest algorithm. Minority instances are weighted higher (Class 1), while majority instances are weighted lower (Class 0).

3.4.2. Voting (Classification)

Similar to the traditional Random Forest, each tree predicts the class label of an instance, and the final prediction is obtained through majority voting (classification). Instead of using the standard Gini impurity or mean squared error in each decision tree node to evaluate the quality of a split, the algorithm uses the weighted version of these metrics.

3.4.3. Classification Tasks

Weighted Gini Impurity (Gini Index):

$$Gini\_weighted = w\_0 * Gini(Class\ 0) + w\_1 * Gini(Class\ 1) \quad (1)$$

Where Gini(Class i) is the Gini impurity of class i (i.e., the probability of misclassifying a randomly chosen element from Class i).

3.4.4. Regression Tasks

Weighted Mean Squared Error (MSE):

$$MSE\_weighted = w\_0 * MSE(Class\ 0) + w\_1 * MSE(Class\ 1) \quad (2)$$

Where MSE(Class i) is the mean squared error of Class i.

3.4.5. Class Weights

The class weights are assigned so minority instances are weighted higher, while majority instances are weighted lower. The exact weighting scheme can vary, but a common approach is to set the weight of each class inversely proportional to its frequency in the dataset. Random Forest ensembles make predictions during the testing phase. Decision trees are voted upon by a majority in case of classification. In regression, the final prediction is the average of the predictions from all trees.

The class weights w_0 and w_1 are usually calculated based on the class frequencies in the training dataset. One common approach is to set the class weights inversely proportional to their frequencies. For example, Mathematically, let's define the weighted voting for Class 1 (minority class) as V1 and Class 0 (majority class) as V0.

$$V1 = \Sigma\,(1\{Class\ 1\ prediction\} * w1) \,/\, \Sigma\, w1 \quad (3)$$

$$V0 = \Sigma\,(1\{Class\ 0\ prediction\} * w0) \,/\, \Sigma\, w0 \quad (4)$$

Where, Among the N instances in this dataset, n0 instances belong to Class 0, and n1 instances belong to Class 1. The class weights are represented as w0 (weight for Class 0) and w1 (weight for Class 1), where w1 > w0.

1{Class 1 prediction} is an indicator function that equals 1 if the tree's prediction is Class 1 and 0 otherwise. 1{Class 0 prediction} is an indicator function that equals 1 if the tree's prediction is Class 0 and 0 otherwise. Σ represents the sum of all decision trees in the ensemble. The final prediction is then determined based on comparing V1 and V0: If V1 > V0, the instance is classified as Class 1. If V0 > V1, the instance is classified as Class 0.

The weighted class Random Forest inherits robustness, accuracy, and high-dimensionality handling abilities from the traditional RF algorithm. Weighted class Random Forest is particularly useful in scenarios where the class imbalance is prevalent, such as in a multi-source social media environment.

Pseudocode for MRF Classifier

Input:
- Number of decision trees in the forest (num_trees).
- Class weights for each class (class_weights).

Output:
- Weighted Class Random Forest model, a collection of decision trees.

```
def weighted_class_random_forest(data, labels, n_trees, n_features):
  predictions = []
  for _ in range(n_trees):
    tree = create_weighted_class_random_tree(data, labels, n_features)
    prediction = tree.predict(data)
    predictions.append(prediction)
  return predictions
def create_weighted_class_random_tree(data, labels, n_features):
  tree = DecisionTreeClassifier()
  class_weights = calculate_class_weights(labels)
  for _ in range(n_features):
    feature = random.randint(0, len(data[0]) - 1)
    tree.add_feature(feature, class_weights)
  tree.fit(data, labels)
  return tree
def calculate_class_weights(labels):
  class_counts = np.unique(labels, return_counts=True)
  class_weights = class_counts[1] / np.sum(class_counts[1])
  return class_weights
```

The main steps of the above pseudocode are: Calculate the class weights for the given labels. Create a random forest tree using the weighted class weights. Repeat the process of creating trees until the desired number of trees is created. Average the predictions of the trees to get the final prediction of the classifier.

4. Results and Discussions

In machine learning, model performance evaluation is typically measured using various metrics that quantify the model's accuracy, precision, recall, and F1 score as important aspects. Below are the formulas for some commonly used evaluation metrics:

4.1. Accuracy (ACC)

As a percentage, accuracy represents the number of correctly classified instances.

$$Accuracy = (Number\ of\ Correctly\ Classified\ Instances) / (Total\ Number\ of\ Instances) \quad (5)$$

4.2. Precision (P)

Out of all instances predicted as positive, precision measures the proportion of true positives. This study focuses on predicting positive outcomes accurately.

$$Precision\ (P) = TP / (TP + FP) \quad (6)$$

4.3. Recall (R)

It is calculated by dividing the total number of positive records that are actually true by the number of positive records that are actually true. Coverage of positive cases is a focus area given a lot of attention.

$$Recall\ (R) = TP / (TP + FN) \quad (7)$$

4.4. F1 Score (F1)

Precision and recall are summed up to form the F1 score. There are some situations where it can be beneficial to balance precision with recall, especially when dealing with datasets that are imbalanced.

$$F1 - score = 2 * (Precision * Recall) / (Precision + Recall) \quad (8)$$

Binary classification has four possible outcomes: TP, FP, FN, and TN when compared to the true class labels predicted by a machine learning model. Table 1 shows the various metrics used to evaluate the model's performance based on these outcomes.

Table 1. Comparison of possible outcomes

	Decision Tree	MLP	XG Boost	KNN	RF	MRF
TP	60	75	86	92	95	96
FP	14	11	8	6	3	2
FN	16	12	7	5	4	3
TN	11	8	6	4	3	2

When comparing the performance of different machine learning algorithms, it is essential to use multiple evaluation metrics to comprehensively understand their strengths and better results, as given in Table 2.

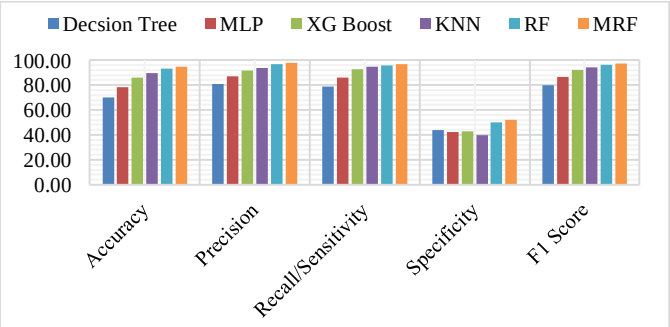

Fig. 4 Performance analysis of proposed model

Table 2. Comparison of performance metrics for different algorithms

	Decision Tree	MLP	XG Boost	KNN	RF	MRF
Accuracy	70.30	78.30	85.98	89.72	93.33	94.62
Precision	81.08	87.21	91.49	93.88	96.94	97.96
Recall/ Sensitivity	78.95	86.21	92.47	94.85	95.96	96.55
Specificity	44.00	42.11	42.86	40.00	50.00	52.13
F1 Score	80.01	86.71	91.98	94.36	96.45	97.56

From the above results in Figure 4, accuracy is evaluated using possible outcomes of the given datasets. The accuracy of the Decision tree is 70.30%, in MLP is 78.30%, in XG Boost is 85.98%, in KNN is 89.72%, traditional RF has 93.33%, and finally Modified Random Forest algorithm provides 94.62%. The results clearly show that the proposed model gives better output than other machine learning models.

5. Conclusion

The weighted class approach effectively addressed the issue of imbalanced class distribution, allowing the model to better capture and classify offensive content, especially in cases where the minority class (offensive content) is significantly underrepresented. The multi-source social media environment provided diverse and heterogeneous data, which further challenged the model's generalization capability. However, the Weighted Class Random Forest algorithm exhibited robustness across different data sources, showcasing its ability to handle variations and complexities in social media content. The MRF algorithm showed notable improvements in performance metrics compared to traditional Random Forest or other classifiers on imbalanced datasets. By assigning higher weights to instances of the minority class, the algorithm achieved higher precision and recall for offensive content identification, enabling more accurate and comprehensive classification results.

In future, Fine-tuning hyperparameters, exploring other ensemble techniques, or integrating advanced Natural Language Processing (NLP) methods could potentially enhance the model's performance further. Additionally, continuous monitoring and updates of the model in response to evolving online content trends are crucial for sustained effectiveness.

References

[1] Mst Shapna Akter et al., "Deep Learning Approach for Classifying the Aggressive Comments on Social Media: Machine Translated Data Vs Real Life Data," *IEEE International Conference on Big Data*, IEEE, pp. 5646-5655, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Hitesh Kumar Sharma, K. Kshitiz, and Shailendra, "NLP and Machine Learning Techniques for Detecting Insulting Comments on Social Networking Platforms," *International Conference on Advances in Computing and Communication Engineering*, IEEE, pp. 265-272, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[3] Mohammed Ali Al-Garadi et al., "Predicting Cyberbullying on Social Media in the Big Data Era Using Machine Learning Algorithms: Review of Literature and Open Challenges," *IEEE Access*, vol. 7, pp. 70701-70718, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Sunil Saumya, Abhinav Kumar, and Jyoti Prakash Sing, "Offensive Language Identification in Dravidian Code Mixed Social Media Text," *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pp. 36-45, 2021. [Google Scholar] [Publisher Link]

[5] Kazi Saeed Alam, Shovan Bhowmik, and Priyo Ranjan Kundu Prosun, "Cyberbullying Detection: An Ensemble Based Machine Learning Approach," *Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks*, IEEE, pp. 710-715, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Fatemah Husain, "Arabic Offensive Language Detection Using Machine Learning and Ensemble Machine Learning Approaches," *arXiv*, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] Byungdae An, and Yongmoo Suh, "Identifying Financial Statement Fraud with Decision Rules Obtained from Modified Random Forest," *Data Technologies and Applications*, vol. 54, no. 2, pp. 235-255, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Dong-Sheng Cao et al., "In Silico Classification of Human Maximum Recommended Daily Dose Based on Modified Random Forest and Substructure Fingerprint," *Analytica chimica acta*, vol. 692, no. 1-2, pp. 50-56, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[9] Robert Bryll, Ricardo Gutierrez-Osuna, and Francis Quek, "Attribute Bagging: Improving Accuracy of Classifier Ensembles by Using Random Feature Subsets," *Pattern Recognition*, vol. 36, no. 6, pp. 1291-1302, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[10] Jasmine Shaikh, and Rupali Patil, "Fake News Detection Using Machine Learning," *IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security*, IEEE, pp. 1-5, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Raju Kumar, and Aruna Bhat, "A Study of Machine Learning-Based Models for Detection, Control, and Mitigation of Cyberbullying in Online Social Media," *International Journal of Information Security*, vol. 21, no. 6, pp. 1409-1431, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Mitushi Raj et al., "An Application to Detect Cyberbullying Using Machine Learning and Deep Learning Techniques," *SN Computer Science*, vol. 3, no. 5, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Shaokang Cai et al., "An Reinforcement Learning-Based Speech Censorship Chatbot System," *The Journal of Supercomputing*, vol. 78, no. 6, pp. 8751-8773, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Amit Sheth, Valerie L. Shalin, and Ugur Kursuncu, "Defining and Detecting Toxicity on Social Media: Context and Knowledge are Key," *Neurocomputing*, vol. 490, pp. 312-318, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Nabil Badri, Ferihane Kboubi, and Anja Habacha Chaibi, "Combining Fasttext and Glove Word Embedding for Offensive and Hate Speech Text Detection," *Procedia Computer Science*, vol. 207, pp. 769-778, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Malliga Subramanian et al., "Offensive Language Detection in Tamil Youtube Comments by Adapters and Cross-Domain Knowledge Transfer," *Computer Speech & Language*, vol. 76, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] Fatima Shannaq et al., "Offensive Language Detection in Arabic Social Networks Using Evolutionary-Based Classifiers Learned from Fine-Tuned Embeddings," *IEEE Access*, vol. 10, pp. 75018-75039, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] Tanmoy Chakraborty, and Sarah Masud, "Nipping in the Bud: Detection, Diffusion and Mitigation of Hate Speech on Social Media," *ACM SIGWEB Newsletter*, pp. 1-9, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Sneha Chinivar et al., "Online Offensive Behaviour in Socialmedia: Detection Approaches, Comprehensive Review and Future Directions," *Entertainment Computing*, vol. 45, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Amit Kumar Balyan et al., "A Hybrid Intrusion Detection Model Using EGA-PSO and Improved Random Forest Method," *Sensors*, vol. 22, no. 16, pp. 1-20, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Caihui Liu et al., "An Improved Decision Tree Algorithm Based on Variable Precision Neighborhood Similarity," *Information Sciences*, vol. 615, pp. 152-166, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] Nasir Jalal et al., "A Novel Improved Random Forest for Text Classification Using Feature Ranking and Optimal Number of Trees," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 2733-2742, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] Tao Wu et al., "Intrusion Detection System Combined Enhanced Random Forest with SMOTE Algorithm," *EURASIP Journal on Advances in Signal Processing*, pp. 1-20, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Hanane Zermane, and Abbes Drardja, "Development of an Efficient Cement Production Monitoring System Based on the Improved Random Forest Algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 120, no. 3-4, pp. 1853-1866, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] Mahsa Hosseinpour et al., "A Hybrid High-Order Type-2 FCM Improved Random Forest Classification Method for Breast Cancer Risk Assessment," *Applied Mathematics and Computation*, vol. 424, 2022. [CrossRef] [Google Scholar] [Publisher Link]